Communicating Sequential Processesの数学モデル(同期の数理)

AUTHOR(S):

萩野, 達也

CITATION:

萩野, 達也. Communicating Sequential Processesの数学モデル(同期の数理). 数理解析研究所講究録 1982, 470: 91-96

ISSUE DATE:

1982-10

URL:

http://hdl.handle.net/2433/103225

数理解析研究所講究録
470巻 1982年 91-96

Communicating Sequential Processes の数学モデル

数理解析研究所　萩野　達也

Communicating Sequential Processes[1]（以下 CSP と略す）の提唱者である C.A.R. Hoare の論文 A Model for Communicating Sequential Processes (On the Construction of Programs, R.M. McKeag and A.M. Macnaghten, Cambridge University Press, 1980 pp.229-254) の内容を紹介する。この論文では、CSP で書かれたプログラムの正当性を証明する際の基礎となる、CSP の数学的モデルを与えている。

1. 記法および基本的概念

まず process は、それが関与した event を表わす記号の有限列（関与した順序に並ぶ）である trace の集合として定義される。process P に対して P は、P の関与できる event の全体を表わす。まず、CSP の単純な構文に対しては、

ABORT = $\{\langle\rangle\}$　　（$\langle\rangle$ は空列）

$c \to P = \{\langle\rangle\} \cup \{\langle c\rangle s \mid s \in P\}$

$P \,[]\, Q = P \cup Q$ †

と定義する。process の定義は、しばしば再帰的に行われる。これは、BNF文法と同様に取扱う。

2. 複合文

2つのprocess P と Q の複合実行を取り扱う。まず、並列に実行する場合には、互に共通に関与する event は同期をとらないといけないので

$$P \| Q = \{ s \in (\bar{P} \cup \bar{Q})^* \mid (s \restriction \bar{P}) \in P \text{ かつ } (s \restriction \bar{Q}) \in Q \}$$

となる。ここで、X^* は X の元からなる有限列全体の集合を、$s \restriction X$ は列 s より X 以外の元を取除いた列を、それぞれ表わす。

次に、P の実行の後に Q を行う場合を考える。そのために、特別な記号 ✓ を導入する。✓ はすべての process に共通な event であり、process が（successfully に）停止したことを示す。†

$$P ; Q = \{ s \in P \mid s \text{ は } \surd \text{ を含まない} \} \cup \{ st \mid s\langle\surd\rangle \in P , t \in Q \}$$

program らしくする（✓ を program 中に書かない）ために、process SKIP $= \{\langle\rangle, \langle\surd\rangle\}$ と定義し、$d \to$ SKIP を d と略する。

3. Alphabet Transformation

Alphabet Transformation f は、記号の変換で、process P に対して f(P) は、trace 中の記号すべてに f を適用してえられ

た（ trace をもつ ）process を表わす。重要な例として、process の名前付けがある。m と名前付けられた process P は、Alphabet Transformation $prefix_m(x)=m.x$ によって、m:P = $prefix_m(P)$ と定義される。

次に、局所的な event を外部から隠すために、

$$[P \| Q] = (P \| Q) \setminus (\bar{P} \cap \bar{Q} - \{\surd\})$$

と定義する。† ここで、$P \setminus X$ は $\{s \setminus (\bar{P}-X) \mid s \in P\}$ である。

4. 入出力

まずはじめに、名前なしの入出力を定義する。!t, ?t は、それぞれ t の出力、入力を示す event である。入力に対する program を書きやすくするために、

$$?x:T \to P_x = \{\langle\rangle\} \cup \{\langle ?t \rangle s \mid t \in T, s \in P_t\}$$

と定義する。T は入力の型で、x はダミー変数である。

process P の出力をすべて process Q に送る場合には、

$$P \gg Q = [strip!(P) \| strip?(Q)]$$

となる。ここで、strip!(x) は x が !t なら ! を取除き t とする Alphabet Transformation である。strip? も同様。これによって、P の出力と Q の入力は同一 event となり、同期がとられる。

次に、名前付き process との入出力は、同様に、m!t(m?t)

を、mと名前付けられたprocessに出力（から入力）tを送る（受取る）ことを表わすeventとする。同期をとるために、$prefix_m$ の定義に、

$$prefix_m(?t)=m!t \qquad prefix_m(!t)=m?t$$

$$prefix_m(n?t)=prefix_n(m!t)=n.m!t$$

を付け加える。n.m!t は process n から process m へ t が通信されたことを表わす event である。

5. Sharing と例

program を簡単にするために、

$$[\|\ x:X]P_x = P_u \| P_v \| \cdots \qquad X=\{u,v,\cdots\}$$

と定義する。$[][x:X]P_x$ も同様に定義する。†

以上のような定義をもとに、この論文では、A Multiprogrammed Batch Processing System の記述例をあげている。これは、複数の card reader からの job 入力を、複数の processor で処理を行い、その結果を複数の line printer に出力する systemである。

6. 離散事象の Simulation

最後に、離散事象の simulation を行うモデルを取扱う。ここでは、process は現在の時間を知ったり、ある時間になる

まで動作を停止することができる。計時 process は、

$$TIM_t = ([]x:ANY]x!t \rightarrow TIM_t$$
$$[][]x:ANY]x?t \rightarrow TIM_t$$
$$[]otherwise \rightarrow TIM_{next(t)})$$

と定義されるが、otherwise という event は、他の event が起こらない時にのみ実行しないと、ある process を起動し忘れたりする。そのために、$rescue_e(P)$ を定義する。

$$rescue_e(P) = Q \setminus \{e\}$$
$$Q = \{s \in P \mid s = t\langle e\rangle u \text{ かつ } t\langle x\rangle \in P \text{ ならば } x = e\}$$

これをもちいると、計時 process を使う process USERS は、

$$simulate(USERS) = rescue_{timer.otherwise}(timer:TIM_0 \,||\, USERS)$$

となる。

7. おわりに

この論文では、CSP に数学的モデルを与えているが、形式化は最小にとめられており、証明もすべて省かれている。

最後に、いくつかの注釈と疑問をあげる。

1) いくつかの非決定性を取除くために、論文の最後にあげられてある制限が加えられている。(本紹介では † で印をつけた。)

2) 再帰的な定義に対して、もうすこし考察する必要があるのではないか。

3) process は、途中状態の trace も含む集合として定義されているが、最終結果だけではいけないのか。それによって、✓はいらなくならないのか。

4) 名前なし process, 入出力があるために、従来の CSP より使いやすい。さらに、計時 process 等では、ANY という、process 全体の集合をもちいている。

5) 離散事象の simulation で、repeat timer?x until x=8 のような busy wait はどうなるのか。

6) このモデルでは、通信、同期のみを取扱っており、代入文、if 文、while 文の取扱いはどうするのか。

10. 参考文献

1) C.A.R.Hoare,Communicating Sequential Processes. CACM 21,8,1978.pp.666-677.

2) N.Francez,C.A.R.Hoare,D.J.Lehmann and W.P.de Roever, Semantics of Nondeterminism,Concurrency and Communication. JCSS 19,1979.pp.290-308.

3) K.R.Apt,N.Francez and W.P.de Roever,A Proof System for Communicating Sequential Processes.ACM TOPLAS 2,3,1980. pp.359-385.

4) G.M.Levin and D.Gries,A Proof Technique for Communicating Sequential Processes.Acta Informatica 15,1981.pp.281-302.